Priority assignment on an avionics switched Ethernet Network (QoS AFDX)

Tasnim Hamza, Jean-Luc Scharbarg, Christian Fraboul
Université de Toulouse - IRIT/ENSEEIHT/INPT
2, rue Camichel - 31000 Toulouse - France
Email: firstnanme.lastname@enseeiht.fr

***Abstract*—AFDX (Avionics Full Duplex Switched Ethernet) standardised as ARINC 664 is a major upgrade for avionics systems. For current aircrafts, it implements a FIFO scheduling policy and allows the transmission of sporadic flows between avionics functions distributed on a set of end systems. The certification imposes to guarantee that the end-to-end delay of any frame transmitted on the network is upper-bounded and that no frame is lost due to buffer overflow. This guarantee is obtained thanks to a worst-case analysis which is based on either network calculus or trajectory approach. However it leads to an over-dimensioning of the network.**

For future aircraft, it is envisioned to use a Fixed Priority scheduling policy in order to better use network resources (QoS AFDX). Existing AFDX switches implement two priority levels. A worst-case analysis of such a network exists, based on the Trajectory approach. Thus, the remaining issue is to assign efficiently the available priorities to the flows.

The contribution of this paper deals with this issue. It proposes to assign the priorities to the flows using the well-know Optimal Priority Assignment algorithm (OPA) which was first defined for monoprocessor preemptive systems. The proposed solution is applied on two case studies. The overall worst-case delay is reduced by 30 % on a small configuration and 20 % on a realistic one.

I. Introduction

Designing and manufacturing new civilian aircrafts has lead to an increase of the number of embedded systems and functions. The AFDX [1] brings an answer by multiplexing a huge amount of communication flows over a full duplex switched Ethernet network. It has become the reference communication technology in the context of civilian avionics and provides a backbone network for the avionics platform.

Since flows are multicast, each frame of a given flow follows a set of paths. It experiences a delay along each path. This delay is the sum of transmission delays on links and latencies in switches. There are no collisions on full duplex links. Thus transmission delays are directly derived from link rates and frame lengths. Latencies in switches mainly depend on the waiting times in output queues. Indeed each output port is shared by a set of flows. Since AFDX flows are asynchronous, frames from competing flows can arrive at any time in an output port. Their order of transmission is determined by the scheduling policy.

AFDX switches implement a fixed priority scheduling at each output port, with two priority levels. For current aircraft, one single priority level is used. It comes to consider a first come first served (FCFS or FIFO) scheduling policy.

For certification purpose, it is mandatory to upper bound the end-to-end delay of all the frames transmitted on the AFDX. Three main approaches have been proposed for this worst-case analysis in the context of FCFS scheduling. The network calculus approach provides sure upper bounds [2] and it has been used for the certification of the AFDX on board of the A380 and A350. The trajectory approach provides tighter sure upper bounds [2]. It has been shown [3] that, for some corner cases, this approach introduces some optimism. However these cases never occur in existing AFDX configurations. The last approach is based on model checking. It gives exact worst-case delays on limited configurations, due to the well-known combinatorial explosion problem [4].

Results show that the worst-case delay can be very different for different flows. On a typical aircraft configuration, it varies from 2 to 15 ms, depending on the considered flow. Actually, it is impacted by the distribution of the flows on the available paths. This variation is an issue when the constraint on the delay is the same for all the flows of the configuration, which is a classical situation. Obviously this worst-case delay constraint cannot be less than 15 ms. If it appends to be less than this value, then, there exist several solutions.

1) Reduce the number of flows: most of the time, this is impossible, due to applicative constraints.
2) Use extra switches and links: it is often too costly.
3) find another static routing which reduces the overall worst-case delay: the idea is to better distribute the flows among the available paths.
4) use the two available priority levels: the goal is to limit the delay variation, thus the overall worst-case delay.

Solutions 3 and 4 are the most interesting ones, since they have no impact on the application nor on the networks architecture.

In this paper, we consider solution 4. Two problems have to be addressed:

- allocate the priorities to the flows
- upper-bound the delays of these flows with priorities.

A solution to the second problem (worst-case analysis) has been proposed in [5]. It is based on the trajectory approach. It has been shown that it gives tight upper bound on delays for flows with priorities transmitted on an AFDX network.

The allocation of priorities to flows transmitted on an AFDX network is the topic of this paper. The problem of priority allocation has been widely studied in the context of

real-time systems. An Optimal Priority Assignment algorithm (OPA) has been proposed in the context of monoprocessor preemptive Systems [6]. This algorithm has been extended to other types of systems (e.g. multiprocessor systems) [7]. OPA is based on a schedulability test which has to respect a set of conditions.

The contribution of this paper is to show that OPA can be extended to the context of an AFDX network implementing a fixed priority scheduling. The schedulability test is based on the Trajectory approach. Then we show that this priority assignment algorithm can be used in order to significantly reduce the overall worst-case end-to-end delay on the network.

The paper is organised as follows. Section II gives the key features of the AFDX network and summarised the Trajectory approach in the context of Fixed Priority scheduling. Section III describes the Optimal Priority Assignment algorithm and shows how it can be applied in the context of AFDX. Section IV presents results on two case studies. Section V concludes the paper and gives some directions for future work.

II. CONTEXT

This section gives the key features of the AFDX network and summarises the worst-case delay analysis of the AFDX based on the Trajectory approach for Fixed Priority scheduling.

A. AFDX network and traffic model

AFDX (Avionics Full Duplex Switched Ethernet), standardised as ARINC 664 [1], has been tailored in order to take into account avionics constrains. Avionics function are distributed on a set of End Systems which are interconnected by a full Duplex switched Ethernet network. An illustrative example is depicted in Figure 1. It is composed of 7 End Systems (ES) (e_1 to e_7) and 3 switches (S_1 to S_3). Each switch uses a store and forward pattern. Switches have no buffers on input ports and buffers on output ports. Each switch has a switching latency bounded by a constant value $L_{max} = 16\mu s$. Each ES is connected to exactly one switch port and each switch port is connected to at most one ES.

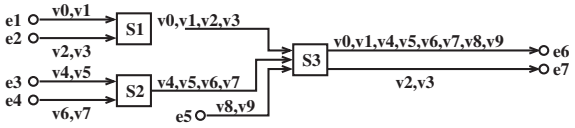

Fig. 1. Illustrative AFDX configuration

End Systems exchange frames through Virtual Links (VLs). A VL statically defines a unidirectional virtual communication channel between one source ES and one or more destination ESs. For instance, in Figure 1, VL $v1$ is defined by the path $\{e1$ - $S1$ - $S3$ - $e6\}$. The definition of a VL includes also the Bandwidth Allocation Gap (BAG) and the minimum and the maximum frame length (S_{min} and S_{max}). BAG is the minimum delay between two consecutive frames of the associated VL (a VL is a sporadic flow). Table I gives BAGs and frame lengths of the ten VLs of the configuration in Figure 1.

A combined Fixed Priority and First In, First Out (FP/FIFO) scheduling policy is implemented in each AFDX switch output port. Up to now, all the VLs are assigned the same priority, leading to a FIFO scheduling in existing AFDX switch.

	v_0, v_1, v_2, v_3	v_4, v_5, v_6, v_7, v_8, v_9
BAG (ms)	4	4
S_{max} (bytes)	500	1000

TABLE I. VL FEATURES

An avionics network has to be certified. Thus, it is mandatory to prove that the End-To-End delay is upper bounded for each flow transmitted on the network. Moreover, the output buffers have to be dimensioned so that no frame is lost due to buffer overflow.

Many work has been devoted to the worst-case analysis of an AFDX network. Up to now, the computation of an exact worst-case End-to-End delay is not possible in a reasonable time on an industrial size configuration (roughly 1000 flows) [4]. Conversely, the computation of a sure upper bound of this End-To-End delay in the context of FIFO scheduling can be obtained by at least two approaches, namely Network Calculus [8] and Trajectory approach [2][9]. The latest approach can also cope with FP/FIFO scheduling [10].

The goal of the work presented in this paper is to assign priorities to flows such that timing constrains of flows are respected. The verification of this timing constrains is based on the trajectory approach with FP/FIFO scheduling. This approach is summarised in the next paragraph.

B. Worst-case analysis based on Trajectory approach

The trajectory approach [11] has been developed to get deterministic upper bounds on end-to-end (ETE) response time in distributed systems. It identifies for a packet *m* belonging to a flow τ_i the worst-case scenario on its trajectory. This approach has been applied and optimised in the context of AFDX network with FIFO output port policy [2][9]. It has been extended to the case of QoS AFDX with fixed priority (FP)/FIFO output port policy [10] based on existing results for combined FP and FIFO [12]. This last extension of the approach is now summarized and illustrated on the example in Figure 1.

The trajectory of a frame includes a set of output ports (including the links) and a set of switch fabrics. In the context of AFDX, the delay of a switch fabric is bounded, while the delay in an output port highly depends on the competing frames. Indeed, depending on the scheduling policy implemented in the nodes, a given frame is delayed by the transmission of the frames with a higher priority and the frames with the same priority which have arrived in the output port before the considered frame.

Let us consider a VL v_i ($i \in [1, n]$) following a path $\mathcal{P}_i$. The frame f_i of VL v_i generated at time t is under study. v_i is characterised by its period BAG_i and its processing time $C_i = S_{max_i}/R$, where S_{max_i} is the maximum size of frames of v_i and $R = 100Mb/s$ is the AFDX link rate (all the AFDX links work at the same rate). The VL v_i is emitted by the source node $first_i$ and it follows a statically defined path $\mathcal{P}_i$. Its last visited node is denoted $last_i$. On a QoS AFDX, v_i is

assigned a priority p_i. The set of VLs having a fixed priority level strictly higher, equal or strictly lower than this of v_i are denoted resp. hp_i, sp_i and lp_i.

In order to compute the worst-case end-to-end delay of frame f_i, the Trajectory approach considers the longest possible busy period ending with frame f_i at each node in the path $\mathcal{P}_i$. Such a busy period $bp(h)$ is a temporal interval with no idle time in node h. The busy periods along the path $\mathcal{P}_3 = \{e_2, S_2, S_3\}$ of v_3 in Figure 1 are illustrated in Figure 2.

Let $f(h)$ denote the first frame of bp^h which crosses h and comes from the node preceding h in $\mathcal{P}_i$. The starting instant of the busy period $bp(h)$ i.e. the arrival time of $f(h)$ is denoted $M(h,i)$, while $a(h,f_i)$ denotes the arrival time of the studied frame f_i at the node h.

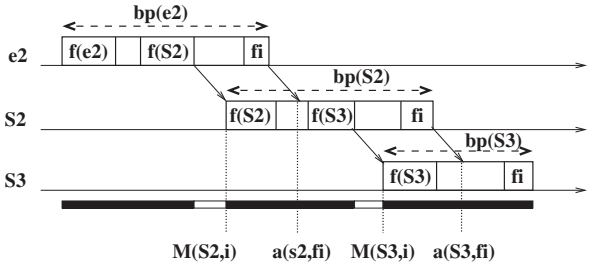

Fig. 2. Busy periods of v_3

Frames from competing VLs generated by other source nodes and crossing the VL v_i will delay its frames.

For example, frames of VLs having the same priority level as v_i, (i.e $\in sp_i$) arriving no earlier than $M(h,i)$ and no later than $a(h,f_i)$ delay the frame f_i . The maximum waiting delay of f_i generated by competing VLs $\in sp_i$ in a busy period bp^h is obtained by maximising the number of frames of these VLs joining v_i in the interval $[M(h,i), a(h,f_i)]$. Frames with higher priorities are also considered, as it will be shown later. This process propagates till the last node.

The computation of the worst-case ETE delay of any VL v_i according to the Trajectory approach for FP/FIFO is bounded by:

$$R_i = \max_{t \geq 0}(W_{i,t}^{last_i} + C_i - t) \tag{1}$$

$last_i$ is the last node of flow τ_i and $W_{i,t}^{last_i}$ is an upper bound on the latest starting time of a frame *m* generated at time *t* on its last visited node. The definition of $W_{i,t}^{last_i}$ given in [10] is:

$$W_{i,t}^{last_i} = \sum_{\substack{j \in sp_i \cup \{i\} \\ \mathcal{P}_j \cap \mathcal{P}_i \neq \emptyset}} \left(1 + \lfloor \frac{t + A_{i,j}}{T_j} \rfloor\right).C_j \tag{2}$$

$$+ \sum_{\substack{j \in hp_i \\ \mathcal{P}_j \cap \mathcal{P}_i \neq \emptyset}} \left(1 + \lfloor \frac{W_{i,t}^{last_{i,j}} + B_{i,j}}{T_j} \rfloor\right).C_j \tag{3}$$

$$+ \sum_{\substack{h \in \mathcal{P}_i \\ h \neq last_i}} \left(\max_{\substack{h \in hp_i \cup sp_i \cup [i] \\ h \in \mathcal{P}_j}} \{C_j\} \right) \tag{4}$$

$$+ (|\mathcal{P}_i| - 1).L_{max} \tag{5}$$

$$+ \sum_{h \in \mathcal{P}_i} \delta_i^h \tag{6}$$

$$+ \sum_{\substack{h \in \mathcal{P}_i \\ h \neq first_i}} \Delta_h \tag{7}$$

$$- C_i \tag{8}$$

Let's have a look at each term of this definition.

Term (2) represents the workload of all the competing VLs with the same priority as the one of v_i under study. The principle is to determine the maximum number of frames of each of those competing VLs which can delay frame f_i. Let us consider a VL v_j which joins v_i in a node $h = first_{i,j}$. v_j is assigned the same priority as v_i. A frame of v_j can delay f_i if it arrives in $first_{i,j}$ during the busy period where f_i is transmitted from $first_{i,j}$ and no latter than the arrival of f_i at $first_{i,j}$. Since the transmission of a frame of VL v_j from its source node $first_j$ to the node $first_{j,i}$ requires at least $S_{min_j}^{first_{j,i}}$ and at most $S_{max_j}^{first_{j,i}}$, the earliest generation time of a frame of VL v_j at its source node must be $M(first_{i,j}, i) - S_{max_j}^{first_{j,i}} - J_j$ and the latest generation time must be $t + S_{max_i}^{first_{j,i}} - S_{min_j}^{first_{j,i}}$. Thus we have:

$$\begin{aligned} A_{i,j} &= S_{max_i}^{first_{j,i}} - S_{min_j}^{first_{j,i}} \\ &\quad - M(first_{i,j}, i) + S_{max_j}^{first_{i,j}} + J_j \end{aligned} \tag{9}$$

Term (3) represents the workload of all the competing VLs with higher priority than v_i. The difference with the previous term is that, due to FP scheduling, a frame from a competing VL can delay f_i in h even if it arrives at h after $a(h,f_i)$. Thus the computation considers $W_{i,t}^{last_{i,j}}$, the latest starting time of f_i at its last shared node $last_{i,j}$ with v_j

According to [11], $f(h+1)$ has to be counted twice in each node, except the slowest one. In AFDX context, the slowest node is arbitrary chosen as the last one [10] since all nodes work at the same speed. In order to guarantee the end-to-end delay upper bound, the frames counted twice are those with the largest size. Thus, this part of the delay is given by:

$$\sum_{\substack{h \in \mathcal{P}_i \\ h \neq last_i}} \left(\max_{\substack{h \in hp_i \cup sp_i \cup [i] \\ h \in \mathcal{P}_j}} \{C_j\} \right) \tag{10}$$

Due to physical constraint, for a frame f_i visiting $|\mathcal{P}_i| - 1$ switches, there is a switch-dependent delay L_{max} computed as follows:

$$\left(|\mathcal{P}_i| - 1\right).L_{max} \tag{11}$$

The trajectory computation also integrates the non-preemption effect. Actually, when a frame arrives in the output port it has to wait until the end of the frame under transmission in this port, regardless of the priority of the frame. Thus, with FP/FIFO policy, f_i can be delayed at most by one lower priority frame. If we consider that at each node h of $\mathcal{P}_i$, f_i is delayed by the frame $\in lp_i$ with the maximum size (denoted δ_i^h) the delay due to non-preemption is bounded by:

$$\sum_{N_h \in \mathcal{P}_i} \delta_i^h \tag{12}$$

In [2] an optimisation of the trajectory approach is proposed in order to take into account the fact that frames transmitted by the same input link are serialised and they cannot arrive at an output port at the same time. Therefore, a serialisation factor is subtracted from the computation. This factor sums for each node h in $\mathcal{P}_i$ the duration between the beginning of bp^h and the arrival of the first frame coming from the preceding node in $\mathcal{P}_i$, i.e. $h-1$. It is computed by:

$$\sum_{\substack{h \in \mathcal{P}_i \\ h \neq first_i}} \Delta_h \tag{13}$$

In [10], Bauer and al. establish a lower bound on $\Delta_{N_h} \forall N_h \in P_i$ in the context of AFDX with FP/FIFO policy. The serialisation term is lower bounded by the maximum of 0 and

$$\max_{1 \leq x \leq k_h} \left(\min(l_x^h)\right) - \max(l_0^h)$$

Now, We illustrate the trajectory approach computation on the sample AFDX configuration depicted in Figure 1. It includes seven ES (e_1 to e_7), three switches (S_1 to S_3) and ten VLs (v_0 to v_9). VL features (BAGs and frame lengths) are summarized in Table I. We assume that v_0, v_1, v_2, v_3 have the lowest priority (1) while the six other VLs have the highest priority (2). Every link works at 100 Mb/s and the switching delay is 16 μs. The transmission time of a frame of 500 (resp. 1000) bytes on a link is 40 (resp. 80) μs.

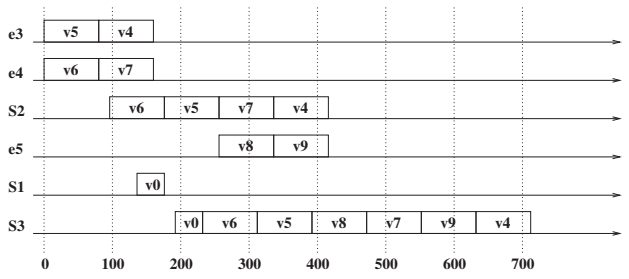

Fig. 3. Worst-case scenario for v_4

Let us analyse the worst-case response time of VLs v_4. The sequence producing worst-case ETE delay with FP/FIFO policy is illustrated in Figure 3. The frame from v_4 is generated at time 0. Let's calculate each term of trajectory approach computation.

Term 2 concerns frames from v_4 and competing VLs with the same priority as v_4. Here, there is one single frame from each VL ($v_4, v_5, v_6, v_7, v_8, v_9$). Thus term 2 is 480 μs.

Term 3 concerns frames from VLs with higher priority than this of v_4. There are no such VLs in the configuration. Thus term 3 is null.

Term 4 represents the extra delay due to frames which have to be counted twice. There is one frame with maximal size for e_4 and S_2, which leads to 160 μs.

Term 5 is the sum of switching latencies (16 μs per switch). v_4 crosses two switches. Thus term 5 is 32 μs.

Term 6 concerns the delay induced by lower priority VLs. There are no such VLs in e_4 and S_2. There are two of them in S_3. They have the same transmission time (40 μs). Thus, term 6 is 40 μs.

Term 7 takes into account the impact of the serialization effect. There is no such impact in our case. Thus term 7 is null.

The worst-case delay of v_4 is the sum of all these terms, which is 712 μs.

III. Assignment of priorities to VLs

The trajectory approach presented in the previous section allows a worst-case delay analysis of an AFDX network implementing a FP/FIFO scheduling of VLs (QoS AFDX). Such a QoS AFDX is envisioned for future aircrafts in order to better use the available resources [10]. The remaining issue of such a network is to efficiently assign the priorities to the VLs.

An Optimal Priority Assignment algorithm (OPA) has been proposed in the context of monoprocessor preemptive Systems [6]. This algorithm has been extended to other types of systems (e.g. multiprocessor systems) [7]. In this paper we show that it can be extended to the context of QoS AFDX.

A. OPA overview

OPA considers a mono-processor preemptive system executing a set of real-time tasks [6]. It has been proved that, for an asynchronous periodic task set, OPA generates an optimal priority ordering while using a polynomial number of schedulability tests [6].

OPA proceeds as follows. It first assigns the lowest priority to one task which respects its deadline with this lowest priority. It continues till the remaining unassigned set of tasks is empty. If at a step no task can be assigned the current priority, no feasible priority assignment exists. An overview of OPA process is given by the following algorithm.

---

for each priority level i, lowest first **do**
  for each unassigned task t **do**
    if t feasible with priority i assuming that all unassigned tasks have higher priority **then**

```
            assign priority i to t;
            break;
        end;
    end;
    if no task is feasible with priority i then
        return unschedulable;
    end;
end;
return schedulable;
```

At each step, a schedulability test is applied to the task which is assigned the current priority in order to determine whether it respects its deadline or not. Let us assume a set of n tasks. At the first step, at most n schedulability tests are applied (n is reached if only one task is feasible with the lowest priority and this task is tested at last). Similarly, at step i, $n-i+1$ tasks are in the remaining set. Thus at most $n-i+1$ schedulability tests are applied. Consequently the maximum number of feasibility tests leading to a priority ordering for n tasks is given by:

$$n + (n-1) + ... + 1 = (n^2 + n)/2$$

The OPA algorithm is applied provided that schedulability test respects the following condition [6] (it is then considered **OPA compatible**):

- *Condition 1:* Schedulability of a task may, according to the test, be dependent on the set of higher priority tasks, but not on their relative priority ordering.
- *Condition 2:* Schedulability of a task may, according to the test, be dependent on the set of lower priority tasks, but not on their relative priority ordering.
- *Condition 3:* When the priorities of any two tasks of adjacent priority are swapped, the task being assigned the higher priority cannot become unschedulable according to the test, if it was previously deemed schedulable at the lower priority.

Let us illustrate OPA algorithm on the example task set in Table II. Classically, each task T_i has a period P_i, deadline D_i, computation time C_i and offset O_i. For sake of simplicity, task offsets are supposed null. We assume that all tasks are independent and that the system is preemptive. Since we have 3 tasks, we consider 3 priority levels {1,2,3} where 1 is the lowest one and 3 is the highest one. The feasibility test used in this example, proposed in [13], consists on the computation of the task response time R_i.

Task	C_i	D_i	P_i
T1	1	5	5
T2	2	10	10
T3	3	15	15

TABLE II. TASK SET SCHEDULABLE WITH OPA

At the first step, OPA looks for one task which can be assigned the lowest priority level 1. The first candidate is $T1$. Assuming that $T1$ is assigned priority 1 while $T2$ and $T3$ are assigned higher priority levels, the worst-case response time of $T1$ is $R_1 = 6$. Since $R_1 > D_1$, $T1$ exceeds its deadline and therefore is not feasible with priority 1. OPA moves then to a new candidate, $T2$. Assuming that $T2$ is assigned priority 1 while $T1$ and $T3$ are assigned higher priority levels, the worst-case response time of $T2$ is $R_2 = 7$. Since $R_2 \leq D_2$, $T2$ respects its deadline and therefore is assigned priority 1. The next step is to assign priority 2. Task $T1$ is the first candidate. Assuming that task $T3$ has a higher priority, $R_1 = 4$. Since $R_1 \leq D_1$, task $T1$ is assigned the priority 2. Finally, we assign the priority 3 to the remaining task $T3$. We find that $R_3 = 3 \leq D_3$ which means that $T3$ is feasible at the priority level 3.

B. Minimising priority levels

The basic OPA algorithm considers that the system supports one priority level per task. Such an assumption might be unrealistic since concrete systems support a limited number of priority levels. In order to overcome this problem, an extension of the OPA algorithm has been proposed in [6]. It allows to minimise the number of priority levels requested for a feasible priority ordering. Actually, a priority is no longer assigned exclusively to one single task; it can be assigned to more than one task.

The minimum number of priority levels is achieved by successively maximising the number of tasks assigned to priority levels from the lowest one to the highest one. This extended version of the algorithm is described as follows.

```
for each priority level i, lowest first do
    for each unassigned task t do
        if t feasible with priority i assuming that all
        unassigned tasks have higher priority then
            assign priority i to t;
        end;
    end;
    if no task is feasible with priority i then
        return unschedulable;
    end;
    if no unassigned task remains then
        break;
    end;
end;
return schedulable;
```

C. Extending OPA to the AFDX case

In this section, we show that OPA algorithm can be used in order to assign priorities to the VLs transmitted on a QoS AFDX network. We consider the extended OPA algorithm which minimises the number of priority levels. Indeed, the number of priority levels available on AFDX switches is low (two in existing switches). The schedulability test is based on the trajectory approach. A deadline is associated to each VL and the worst-case End-To-End delay of each VL is computed by the trajectory approach for FP/FIFO.

It is mandatory to check that the trajectory approach is OPA compatible, i.e. it respects the three conditions given in section III-A. This point is addressed in the following paragraphs.

1) Condition 1: impact of higher priority VLs:
Due to FP policy, the time spent by a frame f_i of VL v_i in a switch depends on the static higher priority frames in the switch that arrives before the start time of the transmission of f_i. The term (3) of the Trajectory computation illustrates this effect.

Obviously, the Trajectory approach takes into account the impact of competing VLs with priority levels higher than this of v_i (i.e. $\in hp_i$). However, this impact do not depend on their relative priority order, since the priority levels are not considered in Term (3). Thus, the trajectory approach meets **condition 1** required for OPA compatibility.

2) Condition 2: impact of lower priority VLs:
The impact of lower priority VLs $\in lp_i$ on the end-to-end delay of frame f_i appears through the non-preemption effect on the AFDX network. It is taken into account in the Trajectory computation by Term (6). Since it computes an upper bound of the end-to-end delay of f_i, the Trajectory approach takes into account a maximum size frames of competing VLs $\in lp_i$ at every visited node in $\mathcal{P}_i$. This latter does not depend of the relative priority ordering of lp_i VLs.

Clearly, the non-preemption effect do not depend on the priority ordering of flows $\in lp_i$. It can be concluded that the trajectory computation meets **Condition 2** for OPA compatibility.

3) Condition 3: variation of R_i as a function of priority levels:
Condition 3 of OPA compatibility imposes the following: If the priority levels of two VLs are swapped, the one being assigned the higher priority must remain feasible if it was initially feasible with the lower priority level. In order to demonstrate that the Trajectory computation meets this condition, we prove that the worst-case end-to-end delay of a given VL v_i, computed with the Trajectory approach, does not increase if a higher priority is assigned to v_i.

Let's illustrate this property with VL v_4 of the configuration in Figure 1. Figures 4 and 5 depict the worst-case delay of v_4 for two different priority assignments. We assume three priority levels.

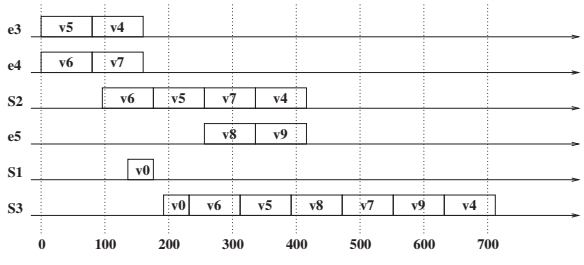

Fig. 4. Worst-case scenario for v_4 with priority 2

In Figure 4, v_4, as well as v_5, $v6$ and v_7 are assigned priority 2 (medium). v_0, v_1, v_2 and v_3 are assigned priority 1 (low) while v_8 and v_9 are assigned priority 3 (high). v_4 is delayed by one frame of each VL with the same or a higher priority and one frame of lower priority (e.g. v_0) in S_3.

In Figure 5, priorities of v_4 and v_8 are swapped. v_4 is delayed by less frames in both S_2 and S_3 (e.g. one single lower priority frame from e_4, i.e. v_6 or v_7). Thus its worst-case delay is reduced.

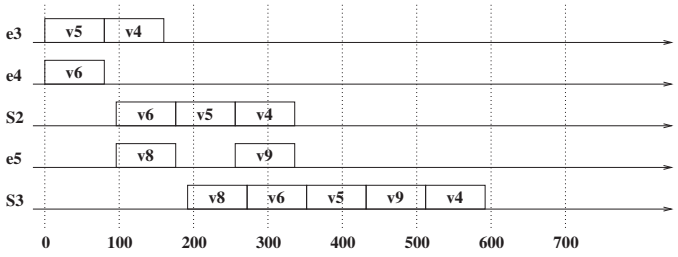

Fig. 5. Worst-case scenario for v_4 with priority 3

Now, we analyse the general case. Let us consider two VLs v_i and v_j to whom was assigned resp. the priorities p_i and p_j ($p_i < p_j$). We suppose that v_i is feasible with priority level p_i, i.e. it respects its deadline according to the trajectory computation. Now, we swap the priorities of v_j and v_i, i.e. v_i is assigned a higher priority level p_j. Let us analyse the impact of the increasing of v_i priority on its end-to-end delay. The terms (5) and (8) are not considered in this analysis given that they do not depend on the priority level of the VLs.

a) Impact of VLs $\in$ $(sp_i \cup hp_i)$: As explained in Sect.II-B, a frame f_i of vi is delayed by frames of competing VLs having the same priority as v_i. This delay is bounded by Term (2) in the trajectory computation, from which we subtract the serialisation (term (7)). f_i is also delayed by frames of competing VLs having a higher priority than that of v_i (Term (3)).

Since p_i ¡ p_j, we have $(sp_j \cup hp_j) \subset hp_i$. Therefore, we have $(sp_j \cup hp_j) \subset (sp_i \cup hp_i)$. That means that with priority level p_j, the VL v_i has a lower number of competing VLs $\in (sp_i \cup hp_i)$. Consequently, the delay induced by VLs with priority level equal or higher than that of v_i decreases.

b) Impact of VLs $\in lp_i$: As explained in Section II-B, due to FP scheduling policy, a frame f_i of vi is delayed by at most one lower priority frame on each node visited by f_i. This delay is illustrated by Term (6) of the Trajectory computation. The frame of lp_i introducing the non-preemption related delay at a given node h is the one having the maximum size among lp_i crossing h. Since $lp_j \supset lp_i$, the maximum size of frame in lp_j might be greater or equal to that in lp_i.

Let's suppose that $\max_{f_k \in lp_j} size(f_k)\} > \max_{f_k \in lp_i} size(f_i)\}$. The frame corresponding to $\max_{f_k \in lp_j} size(f_k)\}$ was counted in Term (2) when v_i was assigned priority p_i. Consequently, assigning priority p_j to v_i reduces Term (2) by the size of this frame and increases Term (6) by $\max_{f_k \in lp_j} size(f_k)\} - \max_{f_k \in lp_i} size(f_i)\}$. Overall, the worst-case delay of v_i is smaller.

IV. Case study

In this section, the proposed solution is evaluated on two case studies. The first one considers a small configuration including 18 VLs. The second one copes with a realistic configuration including 1000 VLs. For both configurations, the goal is to achieve an overall minimisation of the worst-case end-to-end delay of all the flows transmitted on this

AFDX configuration. As explained in the introduction, such a minimisation is relevant in the context of avionics where a classical timing constraint is an upper bound on the delay of any frame transmitted on the network. Paragraph IV-A details the algorithm implemented in order to achieve this goal. Paragraphs IV-B and IV-C gives the network configuration and the results.

A. Implemented algorithm

At the beginning of the process no priority assignment has been done. Thus all the VLs of the considered AFDX configuration have the same priority (it comes to a FIFO scheduling).

In the first step the worst-case end-to-end delay R_i for each VL v_i is computed using the trajectory approach for FIFO. Then we determine the maximum value R_{max} among all these worst-case delays R_i. R_{max} is an upper bound of the end-to-end delay of any frame of any VL transmitted on the network.

In the second step priorities are assigned to the VLs in order to reduce as much as possible this R_{max}. The goal is an overall minimisation of the worst-case end-to-end delay of all the flows transmitted on the network.

More formally, the goal is to find the smallest possible overall worst-case delay R'_{max} such that the priority assignment is feasible. It comes to find the smallest real value x in $[0, 1]$ such that:

- $R'_{max} = x \times R_{max}$
- There exist at least one priority assignment for the VLs of the configuration such that no VL has a worst-case end-to-end delay greater than R'_{max}.

The implementation of this second step consists in decreasing the value of x till the priority assignment becomes not feasible.

A tool implementing this process has been developed in Python. It is based on the tool implementing the trajectory approach for FP/FIFO [10].

B. Case study 1: a small configuration

An overview of the AFDX configuration under study is depicted in the upper part in Figure 6. It includes 7 end systems, 3 switches and 18 VLs. The features of the different VLs are given in Table III.

First the worst-case end-to-end delay for each VL is computed, considering a FIFO scheduling policy. Results are shown in the lower part in Figure 6 (FIFO curve). The maximum worst-case ETE delay R_{max} is that of VL v_{56}. It is equal to 378 μs. This value is taken as the reference in order to define a schedulability condition for the different flows, i.e. the factor x which is applied to R_{max} in order to get the reduced overall maximum end-to-end delay R'_{max}. The smallest value x which leads to a feasible priority assignment is 0.7 (reduction of 30 %). Thus, we have:

$$R'_{max} = 0.7 \times R_{max} = 265\ \mu s$$

The resulting worst-case end-to-end delays for the VLs are shown in the lower part in Figure 6 (OPA curve). The priorities

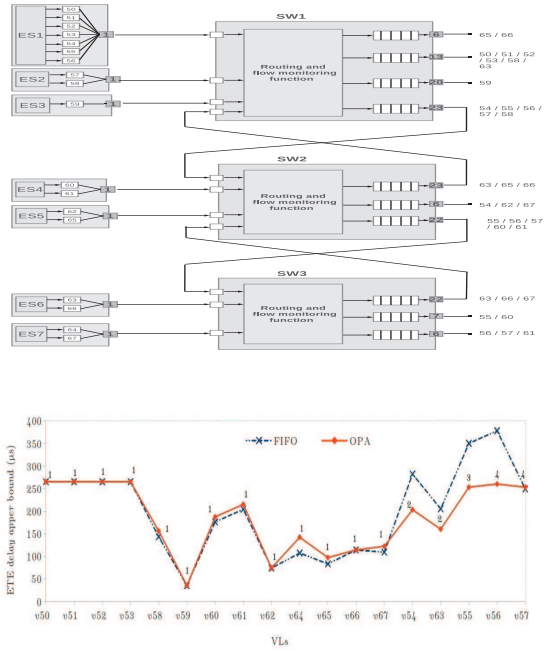

Fig. 6. Sample AFDX configuration

VL_i	BAG (ms)	S_{max}(bytes)
VL_{50}	32	343
VL_{51}	32	171
VL_{52}	16	307
VL_{53}	64	567
VL_{54}	32	183
VL_{55}	128	263
VL_{56}	128	475
VL_{57}	32	217
VL_{58}	16	217
VL_{59}	16	217
VL_{60}	32	343
VL_{61}	64	217
VL_{62}	16	171
VL_{63}	32	217
VL_{64}	32	217
VL_{65}	32	217
VL_{66}	16	171
VL_{67}	16	171

TABLE III. VL FEATURES

assigned to the VLs are shown on the curve. Four priority levels are used by OPA for this configuration. 13 VLS are assigned the lowest priority (1), while respectively 2, 1 and 2 VLs are assigned priority levels 2, 3 and 4.

These results show that OPA can bring a significant reduction on the overall worst-case delay. Next section gives results on a realistic AFDX configuration.

C. Case study 2: a realistic configuration

A general overview of the configuration considered in this case study is depicted in Figure 7. This configuration has been presented in [14]. It includes 123 end systems, 18 switches, roughly 1000 Virtual Links and 6400 VL paths per network (due to VL multicast characteristics). BAGs are harmonic between 2 and 128 ms.

The goal of this case study is to evaluate the reduction of

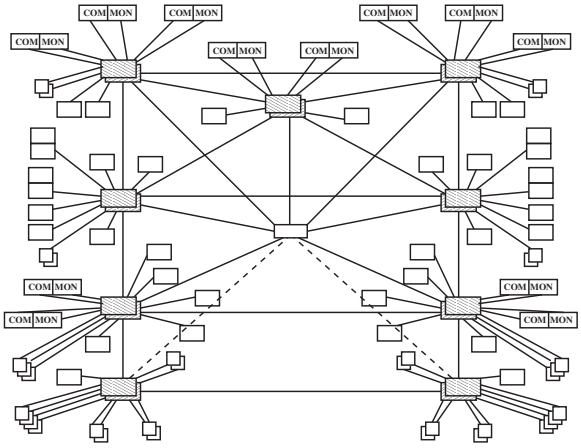

Fig. 7. Realistic AFDX configuration

the maximum worst-case delay provided by the utilization of the two priority levels available in existing AFDX switches.

First the worst-case end-to-end delay for each VL is computed. The maximum worst-case ETE delay R_{max} is 15.413 ms.

The direct application of the proposed algorithm on such a large configuration requires to execute the worst-case delay analysis for each priority assignment to a VL. It means at least 1000 execution of the trajectory computation on the whole network, probably much more since many priority assignments lead to an exceeded delay. In order to manage this amount of computation, it is necessary to minimize the computation time for each trajectory computation. The solution would be to keep all the results of the FIFO computation, then to only modify the values which are impacted by the priority assigned to each VL, in a step by step process. Such an incremental computation is still an open issue.

In a first step, we apply a heuristic. We assign the low priority to the largest possible set of VLs which have the smallest worst-case delay with FIFO (we check that the delay of these VLs do not exceed R'_{max}). Then the high priority is assigned to all the remaining VLs and the R'_{max} constraint is checked on these VLs. The number of execution of the trajectory computation is drastically reduced: one per tested set for the low priority and one at the end. Of course, the drawback is that it can lead to a significantly suboptimal priority assignment.

The application of this heuristic on the configuration in Figure 7 leads to a reduction of 20 % of the overall worst-case delay. 970 VLs out of 1000 are assigned the low priority. This result is very promising.

V. Conclusion

This paper deals with the assignment of priorities to flows transmitted on an avionics switched Ethernet network (QoS AFDX) implementing a fixed priority/FIFO scheduling policy. The proposed solution uses the well-known Optimal Priority Assignment algorithm (OPA) which was first defined for monoprocessor preemptive systems. The schedulability test used by the algorithm is based on the Trajectory approach. Thus we show that this approach is OPA-compatible. Then we show on two case studies is efficient, since the overall worst-case delay of flows is reduced by 30 % for a small configuration and 20 % on a realistic one.

The computation still needs to be optimized in order to be able to cope efficiently with a realistic configuration with roughly 1000 Vls.

The proposed approach considers the worst-case delay as the key constraint of an AFDX network. Another very important feature is the maximum size of buffers needed in each switch in order to guarantee that no frames are lost. It is determined thanks to the computation of the maximum backlog in each switch. Such a computation has been proposed in the FIFO context, based on the Trajectory approach [5]. An extension of this computation with two priority levels has been proposed in [15]. We plan to assign priorities in order to minimise the buffer sizes in the switches.

References

[1] "Arinc specification 664: Aircraft data networks, parts 1,2,7. technical report, aeronotical radio inc." 2002-2005.

[2] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Improving the worst-case delay analysis of an afdx network using an optimized trajectory approach," *IEEE Trans Ind Inf*, vol. 6, no. 4, pp. 521–533, 2010.

[3] G. Kemayo, F. Ridouard, H. Bauer, and P. Richard, "Optimistic problems in the trajectory approach in fifo context," in *18th IEEE International Conference on Emerging Technologies and Factory Automation*. IEEE, sep 2013.

[4] M. Adnan, J.-L. Scharbarg, and C. Fraboul, "Minimizing the search space for computing exact worst-case delays of AFDX periodic flows," in *Proc. of SIES*, Vsteras, june 2011.

[5] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Worst-case backlog evaluation of avionics switched ethernet networks with the trajectory approach," in *Proceedings of the 24th ECRTS*, Pisa, Italy, july 2012.

[6] N. Audsley, "On priority assignment in fixed priority scheduling," *Information Processing Letters*, vol. 79, pp. 39–44, 2001.

[7] R. I. Davis and A. Burns, "Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems," *Real-Time Systems*, vol. 47, no. 1, pp. 1–40, January 2011.

[8] F. Frances, C. Fraboul, and J. Grieu, "Using network calculus to optimize the afdx network," *Proceedings of ERTS*, vol. Toulouse, France, 2006.

[9] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "worst-case end-to-end delay analysis of an avionics afdx network," *Proceedings of DATE, Dresden, Germany*, 2010.

[10] ——, "Applying trajectory approach with static priority queuing for improving the use of available afdx resources," *Real-Time Systems*, vol. 48, no. 1, pp. 101–131, January 2012.

[11] S. Martin and P. Minet, "Schedulability analysis of flows scheduled with fifo: application to the expedited forwarding class," *In: Parallel and distributed processing symposium, IPDPS 2006 20th international*, p. 8, 2006.

[12] ——, "Worst case end-to-end response times of flows scheduled with fp/fifo," *In: ICNICONSMCL'06: Proceedings of the international conference on networking, international conference on systems and international conference on mobile communications and learning technologies. IEEE Computer society, Washington*, p. 54, 2006.

[13] M. Joseph and P. Pandya, "Finding response time in real-time system," *BCS Comp. Jour.*, vol. 29, no. 5, pp. 390–395, 1986.

[14] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul, "Methods for bounding end-to-end delays on an AFDX network," in *Proceedings of the 18th ECRTS*, Dresden, Germany, July 2006, pp. 193–202.

[15] N. R. Garikiparthi, R. Coelho, and G. Fohler, "Calculation of worst case backlog for afdx buffers with two priority levels using trajectory approach," in *RTN*, Paris, France, july 2013.